CHAPTER 9

Spatial data analysis: neighborhood and connectivity operations

In the previous chapter you have seen a number of basic spatial analysis operations used for the overlay of raster maps. Where the overlay operations only consider the combination of raster cells in different maps at the same location, the *neighborhood* operations evaluate the characteristics of an area *surrounding* a specified location. These operations make use of a small calculation window (e.g. of 3 by 3 cells) which does the same calculation for every pixel in the map, taking into account the values of its neighbors. When only the 4 direct neighbors are used (i.e. the ones that are on the same line or the same column as the central pixel), we talk about a 4-connected operation. If all 8 neighbors are taken into account, it is an 8-connected operation (see figure 9.1).

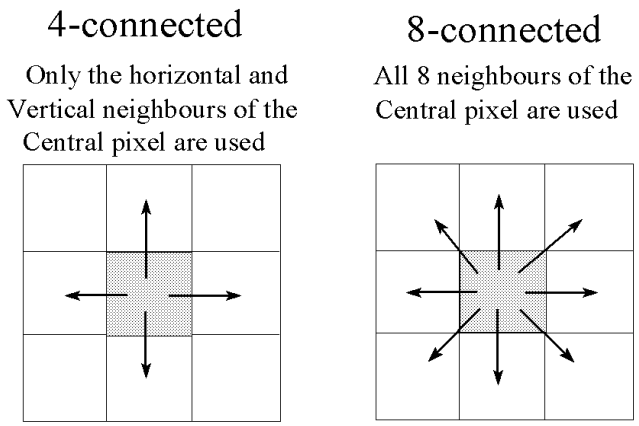


Figure 9.1 The difference between 4-connected and 8-connected neighborhood operations

The calculation window starts with the first pixel on the first line of the map. The result of the calculation is stored in the central pixel. Then the calculation window moves to the second pixel in the first line, and the calculation is repeated. This way the calculation window moves over the entire map. For some of the operations, an iterative procedure is required, in which the calculation window, after finishing the calculation for the last pixel in the last line, will start a reverse operation, for which the input data of the previous run are used. The calculation will be repeated several times until no more changes occur in the map. The following neighborhood operations will be shown in this chapter:

- Filtering (in the previous chapter already explained for satellite images, but now applied to thematic maps). You will look at those filters that are especially used for raster maps, such as smoothing filters, counting filters etc.;
- Neighborhood operators in Map calculation formulas. The use of powerful neighborhood operators and map iterations will be shown, that allow you to generate models that evaluate the conditions of neighboring pixels in a map.;
- Distance calculation, in which the calculation window is used to calculate the distance from source pixels in the map (which could be roads, rivers etc.);
- Area numbering, in which unique identifiers are assigned to groups of pixels with the same class that are connected.

Some other neighborhood operations will be discussed in other chapters:

- Chapter 10 will discuss the specific operations that are used for the analysis using Digital Elevation Models, including contour interpolation and the use of gradient filters in the calculation of slope length and slope direction.
- Chapter 11 will treat the various point interpolation operations available in ILWIS.

In the last part of this chapter some examples are given of how ILWIS can be used for performing connectivity operations. By using these operations, spatial units that are connected (using a set of pre-defined rules) are characterized.

Before you can start with the exercises, you should start up ILWIS and change to the subdirectory c:\ilwis21\data\usrguide\chap09, where the data files for this chapter are stored.

- Double-click the ILWIS program icon in the ILWIS program group.
- Change the working drive and the working directory until you are in the directory c:\ilwis21\data\usrguide\chap09.

9.1 Neighborhood operations: Filtering

In chapter 6, the filter operation has been discussed for the use on (satellite) images. Filtering is a process in which each pixel value in a raster map is replaced with a new value, provided that a condition or a set of conditions is satisfied. The new value is obtained by applying a certain function to each input pixel and its neighbors. There are many types of filters available in ILWIS. Some of these are especially designed for the use on thematic or value maps, instead of satellite images, and some may be used for both.

The following filters are useful to apply on maps:

- **Filters used on Digital Elevation Models**. With the help of the filters working on DEMs, you can calculate slope steepness maps, slope direction maps, slope convexity/concavity, and hillshading. These filters will be treated in the next chapter, which is completely dedicated to the generation and use of DEMs;
- **Majority filters**. These filters allow you to calculate for each pixel the predominant (most frequently occurring) value or class name, of each input pixel and its 8 neighbors (using a 3 by 3 filter);
- **Smoothing filters**. These filters will calculate for each pixel the average value of the central pixel and its 8 neighbors (using a 3 by 3 filter);
- **Rank order filters**. These filters can be used to calculate the median value of the central pixel and its 8 neighbors (using a 3 by 3 filter);
- **Binary filters**. Binary filters regard the input map as a binary map. This means that zero values are regarded as zero, and all other values as one. Depending on the central pixel value and its 8 neighbors, the filter produces a zero or one as output value. These filters are used in what is called *mathematical morphology*. You can use them for *dilation* (the growing of "true" pixels in a certain direction) and *erosion* (the shrinking of "true" pixels in a certain direction);
- **Counting filters**. Counting filters are used to count the number of pixels within a certain search radius around each central pixel. They can be used to count the number of point features, such as houses, wells or landslides.

In the following sections examples will be given of the use of these different filters.

Majority filters

These filters allow you to calculate for each pixel the predominant (most frequently occurring) value or class name, of each input pixel and its neighbors. You can use this filter to simplify a map which has a large variation in classes or values for neighboring pixels.

A very simple example will be used in this exercise: a map showing the ITC logo. It has only 150 lines and 132 column so that you can clearly see the effect of the

filter on the map. The map `Itclogo` is a value map, and contains the values 1 to 3 for showing the logo and the name. The area outside the logo is undefined.

☞

- Display the raster map `Itclogo` with the Pseudo representation.
- Find out the values of the different pixels by clicking them.
- Close the map window.

Now we will apply a *Majority* filter. The majority filter (MAJORITY) is a 3 by 3 filter. As output value for each central pixel, the filter selects the predominant (most frequently occurring) value or class name, of each input pixel and its 8 neighbors. If all 9 pixels have a different value or class, the first value or class name encountered is used as output.

☞

- Click the map `Itclogo` with the right mouse and select Image processing, and the command Filter. The Filtering dialog box is opened.
- Select the Filter type: `Majority`, and the Filter Name: `Majority`.
- Type the Output raster map: `Logo1`. Select Show. Click OK. The map is now calculated.
- Select the representation Pseudo in the Display Options dialog box. Click OK. The map is now displayed.
- Compare the maps `Logo1` and `Itclogo`.
- Close the two map windows.

The result of the majority filter is that the area of the logo is a bit smaller now. The open areas that were within the logo are now filled up, and the text now contains undefined values at the corners (since there the majority of the 8 neighbors is also undefined). There is also a majority filter which is used only on undefined values. The Undef-majority filter (MAJUNDEF) is a 3 by 3 filter which only selects the predominant value or class name if the central pixel is undefined. If the central pixel is not undefined, the value or class name remains the same.

☞

- Filter the map `Itclogo` now with the majority filter Majundef, and create the output map `Logo2`.
- Display all three maps (`Itclogo, Logo1`, and `Logo2`) next to each other and compare them.

The result of the undefined majority filter, is that the undefined pixels which were the neighbors of the pixels of the logo, now also have a value. The logo has increased in size. Similarly to the undefined majority filter, there is also a zero majority filter (MAJZERO), which only selects the predominant value or class name if the central pixel has a zero value. If the central pixel is not zero, the value or class name remains the same.

☞

- Close the three maps (`Itclogo, Logo1`, and `Logo2`).

Smoothing filters

These filters calculate for each pixel the average value of the central pixel and its 8 neighbors (if using a 3 by 3 filter). To see the effect of smoothing filters, we will change the undefined values of the map `Itclogo` into zero values first, before running the filter AVG3X3. This filter calculates the average value of each 9 pixel values considered (3x3 matrix).

☞

- Type the following formula on the command line:
  `Itclogo2:= iff(isundef(Itclogo), 0 , Itclogo)`
- Press Enter. Click OK in the Raster Map Definition dialog box.
- Double-click the map `Itclogo2`, and select the Pseudo representation in the Display Options dialog box. Check whether the undefined values are now zero.
- Filter the map `Itclogo2` with the `Average` filter type, with 3 rows and columns. Create the output map `Logo3` and change the precision to `0.1`
- Display map `Logo3` next to map `ItcLogo2` and compare them.
- Click the pixels in both maps to find out their values.

You can clearly see that the filtered map `Logo3` has a smooth appearance. Whereas the unfiltered map `ItcLogo2` only contains the values 0, 1, 2 and 3, the filtered map has all intermediate values. The smoothing will even be more when we use a larger filter size (say 5 by 5).

☞

- Filter the map `Itclogo2` with the `Average` Filter Type, now with 5 rows and 5 columns.

- Create the output map `Logo4` and change the Precision to `0.1`.
- Display the map `Logo4` with the `pseudo` representation next to the maps `Itclogo2` and `Logo3` and compare them. Click the pixels in both maps to find out their values.
- Close all the maps when you are finished.

Rank order filters

These filters can be used to calculate the median value of the central pixel and its 8 neighbors (if using a 3 by 3 filter).

☞

- Filter the map `Itclogo2` with the `rank order` Filter Type and the filter name `Med3X3`. Create the output map `Logo5`. Do not change the Precision since that will have no effect with a median filter.
- Display the map `Logo5` next to the map `Itclogo2` (both with the pseudo representation) and compare them. Click the pixels in both maps to find out their values.
- Close all maps when you are finished.

The result of this filter is rather similar to the output of the majority filter. The effect can be better seen when we filter a map with more diverse values, such as the `Slope` map.

☞

- Filter the map `Slope` with the rank order filter type and the filter name `Med5X5`. Create the output map `Slopefil`. Do not change the Precision.
- Display the map `Slopefil` next to the map `slope` (both with the pseudo representation) and compare them. Click the pixels in both maps to find out their values.
- Close all maps when you are finished.

Binary filters

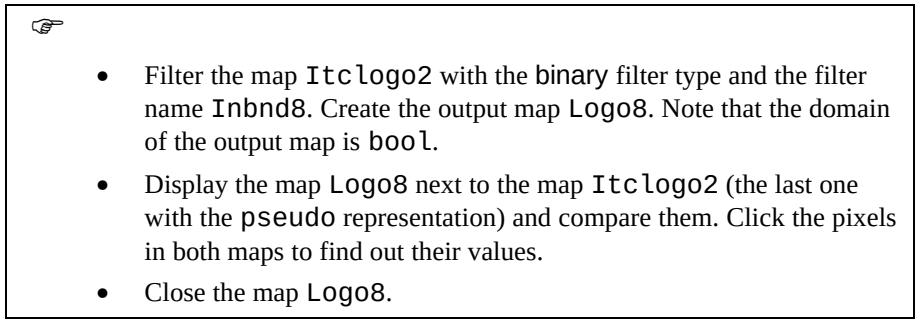
Binary filters regard the input map as a binary map. This means that zero values are regarded as zero and all other values as one. Depending on the central pixel value and its 8 neighbors, the filter produces value zero or one as output values. These filters are used in what is called *mathematical morphology*. You can use them for dilation (the growing of “true” pixels in a certain direction) and erosion (the shrinking of “true” pixels in a certain direction). Let us try some of these filters, using the map `Itclogo2`. First we will use the `Dilate8` filter, which will make the logo grow in size.

- Filter the map `Itclogo2` with the `binary` Filter Type, and the filter name `Dilate8`. Create the output map `Logo6`. Note that the domain of the output map is `bool`.
- Display the map `Logo6` next to the map `Itclogo2` (the last one with the `pseudo` representation) and compare them.
- Click the pixels in both maps to find out their values.
- Close the map windows.

The filter `Dilate8` assigns a 1 to the central pixel if any of the neighbors is true. The result of using `Dilate8` is that true pixels 'grow' in all directions. Now the reverse will be done using the `Shrink8` filter. The shrink filter assigns a 1 only when the center pixel is 1 and all neighbors are also 1. The result of using the `Shrink8` filter is that areas of true pixels 'shrink' along their edges; single lines of true pixels and single true pixels totally surrounded by false pixels disappear. This filter will make the logo shrink in size in all directions. If you use the `Shrink4` filter it will only shrink in horizontal and vertical direction, but not in a diagonal direction.

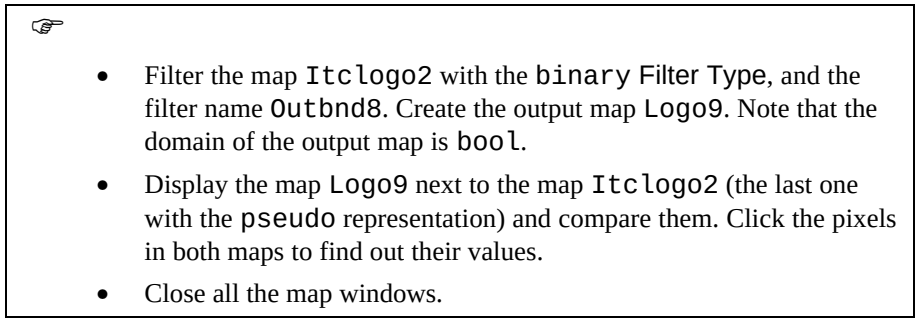
- Filter the map `Itclogo2` with the `binary` Filter Type and the filter name `Shrink8`. Create the output map `Logo7`. Note that the domain of the output map is `bool`.
- Display the map `Logo7` next to the map `Itclogo2` (the last one with the `pseudo` representation) and compare them.
- Click the pixels in both maps to find out their values.
- Close the map `Logo7`.

Next, we will use some filters that provide you with the boundary of units in a map. For example, the filter `Inbnd8` assigns only a 1 when the center pixel is 1 and at least one of the 8 neighbors is 0 (meaning it is located at the edge of a group of true pixels). The result of using this filter is that areas of 8-connected true pixels are replaced by their 8-connected outline (an 8-connected boundary of true pixels).



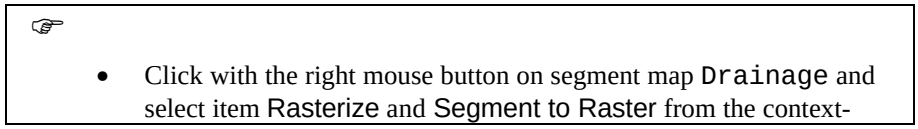
- Filter the map `Itclogo2` with the binary filter type and the filter name `Inbnd8`. Create the output map `Logo8`. Note that the domain of the output map is `bool`.
- Display the map `Logo8` next to the map `Itclogo2` (the last one with the `pseudo` representation) and compare them. Click the pixels in both maps to find out their values.
- Close the map `Logo8`.

The reverse of the `Inbnd8` filter is the `Outbnd8` filter, which gives the outer boundary of areas of 8-connected pixels (3x3 matrix). So instead of giving the outline of the "true" pixels (pixels with a value larger than 0) it will give the outline of the "false" pixels (pixels with a value 0).



- Filter the map `Itclogo2` with the `binary` Filter Type, and the filter name `Outbnd8`. Create the output map `Logo9`. Note that the domain of the output map is `bool`.
- Display the map `Logo9` next to the map `Itclogo2` (the last one with the `pseudo` representation) and compare them. Click the pixels in both maps to find out their values.
- Close all the map windows.

The last of the binary filters which we would like to show you is called the `Conn8to4`. When true pixels are only 8-connected, this filter makes them 4-connected by adding a true pixel in some places. This might give your binary image a better outlook. It is also used to improve line barriers used in the distance calculation. The line barrier must be at least two pixels wide, so that no pixels on either side of the line are connected, not even diagonally. This will be demonstrated using the `Drainage` map.



- Click with the right mouse button on segment map `Drainage` and select item Rasterize and Segment to Raster from the context-

sensitive menu. The Rasterize Segment Map dialog box is opened.

- Select the georeference `Cochabam`. Select Show. Click OK.

  Click OK in the Display Options dialog box. The map is shown.
- Zoom in on a part of the map to see that some of the pixels representing the drainage line have neighbors on both sides that are still 8-connected.
- Filter the map `Drainage` with the `binary` Filter Type and the filter name `Conn8to4`. Create the output map `Draincon`. Note that the domain of the output map is `bool`.
- Display the map `Draincon` next to the map `Drainage`.
- Zoom in on a small section of both maps with the same drainage line and compare them. You can see that the map `Draincon` does not have neighbors on both sides of a drainage line that are 8-connected. The drainage lines are now a real closed barrier.
- Close all map windows.

User-defined linear filters: Example of a counting filter

Besides the standard filters that are stored in the ILWIS system directory, of which we have seen some in the previous part of the exercise, it is also possible to define filters yourself.

One example of a user defined filter is a *counting filter*. Counting filters are used to count the number of pixels, within a certain search radius, around each central pixel. They can be used to count the number of point features, such as houses, wells or landslides. In this part of the exercise we will create a counting filter that will give us the number of landslide pixels within a radius of 10 pixels. The filter will look like this (see next page).

Its form resembles a circle in which all the pixels within the circle (with a radius of 10 pixels) contain the value 1, and the rest the value 0. When we apply this filter on a binary map, showing the landslides, for each pixel the number of pixels with landslides within the 10 pixel radius is calculated.

You will first have to create the binary map with the landslides, using the information from the geomorphologic map (`Geom`).

```
0  0  0  0  0  0  0  0  0  1  1  1  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  1  1  1  1  1  1  1  1  0  0  0  0  0  0  0
0  0  0  0  0  1  1  1  1  1  1  1  1  1  1  1  0  0  0  0  0
0  0  0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  0  0  0  0
0  0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  0  0  0
0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  0  0
0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  0
0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  0
0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  0
1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  0
0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  0
0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  0
0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  0  0
0  0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  0  0  0
0  0  0  0  1  1  1  1  1  1  1  1  1  1  1  1  1  0  0  0  0
0  0  0  0  0  1  1  1  1  1  1  1  1  1  1  1  0  0  0  0  0
0  0  0  0  0  0  0  1  1  1  1  1  1  1  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  1  1  1  0  0  0  0  0  0  0  0  0  0
```

- Type the following formula on the command line: `Landsl:=iff(Geom="al",1,0)`
- Press Enter. The Raster Map Definition dialog box is opened.
- Select the domain `value`, with a range from `0` to `1` and a Precision of `1`. Click OK.
- Display the map `Landsl` and check if the map only contains 0 and 1 values (apart from the undefined values outside of the mapped area). Close the map.
- Click the map `Landsl` with the right mouse button and select Image processing, and the command Filter. The Filtering dialog box is opened.
- Click the Create button next to the list box Filter Name. The Create Filter dialog box is opened.
- Type the Filter Name: `count`, and for the description: `Counting filter`. Change the values for the number of rows and columns both to `21`. Click OK. Now the Edit Filter dialog box is opened.
- Fill in the filter with 0 and 1 values, just as in the example on the previous page.
- Change the Gain to `1.0`.

The gain is the value with which the result of the filter will be multiplied. When a linear filter is placed over the raster map, the filter will multiply all the values of the filter with the corresponding pixel values in the same position in the map. In our case this will result either in a 0 or in a 1. Then all the results of the individual filter cells are summed up and multiplied by the gain. For a counting filter the gain should be 1, since we want to know how many pixels there are with landslides,

within a search radius of 10 pixels around every pixel. The resulting value will be stored in the central pixel, then the filter shifts one pixel further and does the same calculation.

☞

- Click OK in the Edit Filter dialog box. Now you are back in the Filtering dialog box from which you started.
- Type the name of the output map: `Landslnr`.
- Select Show. Note that the value domain is automatically set with the maximum value being the total number of 1 values in your filter.
- Click OK in the Edit Filter dialog box. The map `Landslnr` is now calculated. This will take a while. After that the Display Options dialog box is opened.
- Select the representation `Pseudo` and click OK. The map is displayed.
- Check the values in the map by clicking on the pixels.
- After you have seen the result, close the map window.

Summary: Filter operation using maps

The following filters are useful to apply on maps:

– Filters used on Digital Elevation Models. With the help of the filters working on DEM's you can calculate slope steepness maps, slope direction maps, slope convexity/concavity and hillshading.

– Majority filters. These filters allow you to calculate for each pixel the predominant (most frequently occurring) value or class name of each input pixel and its neighbors.

– Smoothing filters. These filters will calculate for each pixel the average value of the central pixel and its neighbors.

– Rank order filters. These filters can be used to calculate the median value of the central pixel and its neighbors.

– Binary filters. These filters are used in what is called *mathematical morphology*. You can use them for dilation (the growing of "true" pixels in a certain direction) and erosion (the shrinking of "true" pixels in a certain direction).

– Counting filters. Counting filters are used to count the number of pixels within a certain search radius around each central pixel. They can be used to count the number of point features, such as houses, wells or landslides.

9.2 Neighborhood operations using Map Calculation

Neighborhood operations are a special type of spatial analysis in ILWIS. They are calculations on pixels in which the outcome depends on the neighboring pixels. Neighborhood operations may be performed on user-selected pixels as well as on whole maps. Just as in filtering procedures, neighborhood operations make use of a filter. This imaginative window of 3 by 3 cells is moved over the raster map. Each cell of the output map is calculated according to the specified neighborhood expression. The cell numbers in the imaginative window are coded as follows:

1	2	3
4	5	6
7	8	9

Figure 9.2: Neighborhood position identifiers and the neighboring pixels, with respect to the central pixel

This means that the left neighbor of the central pixel is coded number 4 and the lower right pixel number 9. By definition the central pixel itself is included and has a value of 5.

If a neighborhood operation is performed on a pixel on the top or bottom line, or on the very first or last column of a raster map, new neighbors are created by duplicating this boundary line or column. In the case of the neighbor position operator: NBPOS , the outer lines or rows added will have value 0.

The syntax of neighborhood operations is as follows:

```
Map2 = Map1#[Expression]
```

in which:

`Map2`	is the output map
`Map1`	is the input map
`#`	is the neighborhood operator indicating all neighbors are to be used.
`[ ]`	is used to indicate the selection of a specific neighbor.
`Expression`	is any expression with output values in the range 1 to 9 to select a specific neighbor.

In the next example you will calculate the altitude difference in the x and y direction. The `Dx` and `Dy` maps are the output maps using a digital elevation model (`Dem`) as input map.

☞

- Type the following formula on the command line of the main window:

  `dx:=Dem#[4]-Dem#[6]`↵

This formula means: Substract the right neighbor of the central pixel from the left neighbor of the central pixel.

☞

- Press OK in the Raster Map Definition dialog box. The map is calculated.
- Double-click raster map dx. The Display Options dialog box is opened. Select the Representation: pseudo. Click OK.
- Check the result with the pixel information window (using the map Dem and the map Dx).
- Close the pixel information window and the map window.
- Now calculate a map Dy which gives the gradient in the vertical direction.

There are several aggregation functions available for performing operations on neighbors of a pixel. These functions are listed in table 9.1.

Table 9.1: Overview of neighborhood operators

Operator	Explanation
NBMIN (nbexpr)	Returns for each pixel the lowest neighboring value.
NBMAX (nbexpr)	Returns for each pixel the highest neighboring value.
NBSUM (nbexpr)	Returns for each pixel the sum of the neighboring values.
NBPRD (nbexpr)	Returns for each pixel the predominant of the neighboring values.
NBCNDP (nbexpr)	Tests for which of the neighboring pixels a certain condition is true; when none of the neighbors satisfies the specified condition the outcome value is 0.
NBCNT (nbexpr)	Tests the number of neighbors which satisfy the specified condition.
NBFLT (fltname)	Applies a linear filter to all neighboring pixels.
NBDIS	Returns the distance between the central pixel and its neighboring ones.
NBPOS	Returns the position of a neighbor in the neighbor expression.
NBMINP (nbexpr)	Returns the position of the neighbor with the lowest value.
NBMAXP (nbexpr)	Returns the position of the neighbor with the highest value.
NBPRDP (nbexpr)	Returns the position of the neighbor with the predominant value.

Calculating a classified slope direction map

In chapter 10 a method is presented to calculate a slope direction map. A simple classified slope direction map (aspect map) can also be made using the neighborhood operator *nbminp*.

☞

- Type the following formula on the command line of the main

window:

```
Aspect:=nbminp(Dem#) ↵
```

If more than one pixel has the same minimum value, then the precedence of the pixels is determined by the pixel identifier value (see figure 9.2): 5,1,2,3,4,5,6,7,8,9.

The result of the calculation is a raster map with values ranging from 1 to 9 (the position identifiers, see figure 9.2). The value of each pixel is replaced by the position identifier of the neighbor with the lowest value. For example, if the neighbor with the lowest value is the lower left, the outcome of the calculation is 7, if the lowest neighbor is the left, it is 4, etc. The directions are shown in table 9.2.

Table 9.2: The relation between the values of the output map and the direction

Direction	Value
Northwest	1
North	2
Northeast	3
West	4
Flat	5
East	6
Southwest	7
South	8
Southeast	9

☞

- Press OK in the Raster Map Definition dialog box. The map is calculated.
- Double-click the map `Aspect`. The Display Options dialog box is opened. Select the Representation: `pseudo`. Click OK.
- Check the result with the pixel information window.
- Close the pixel information window and the map window.

Calculating a flow direction matrix

One of the examples of the use of neighborhood operators is the calculation of a *flow direction matrix*. If you stand somewhere on the terrain you can calculate in which direction the water will flow away from you (steepest downslope); this can be calculated using the *nbminp* neighborhood operator. With the nbminp you can calculate to which cells the water will flow (the same as the direction calculated in the previous exercise). You can also calculate for each cell, from which of its neighbors it will receive water.

☞

- Type the following formula on the command line of the main window:

  `Flowdir:=10-nbminp(Dem#)` ↵

The result of this calculation is a raster map with values ranging from 1 to 9. The result can be used in other calculations that predict spreading phenomena (such as the spreading of a pollutant, a landslide, or in runoff modeling).The direction matrix value can be compared with the position identification value, in any 3X3 neighborhood of which the cell is part. If the pixel value in the direction matrix equals its position number in any 3x3 neighborhood, then water (or other feature) will flow from this pixel towards the centre of this window for which the statement is true.

☞

- Press OK in the Raster Map Definition dialog box.

  The map is calculated.
- Double-click the map `Flowdir`. The Display Options dialog box is opened. Select the Representation: `pseudo`. Click OK.
- Check the result with the pixel information window.
- Close the pixel information window and the map window.

Determining flat areas and pits in a DEM

An area is considered to be flat when in the moving window of 3 by 3 pixels, all pixels have the same value. With the expression `nbcnt(Dem#=Dem)=9` we can check which pixels have 8 neighbors with the same altitude as the central pixel.

☞

- Type the following formula on the command line of the main window:

  `Flat:=nbcnt(Dem#=Dem)=9` ↵
- Press OK in the Raster Map Definition dialog box.

  The map is calculated.
- Double-click the map `Flat`. The Display Options dialog box is opened.
- Select the Representation: `pseudo`. Click OK.
- Check the result with the pixel information window.
- Close the pixel information window and the map window.

A so-called pit is a pixel which has a value lower than all of its surrounding pixels. We will check the map Dem for the presence of pits. This is done using the following expression:

☞

- Type the following formula on the command line of the main window:

  `Pit:=nbminp(Dem#)=5` ↵
- Press OK in the Raster Map Definition dialog box. Make sure the map has a `bool` domain. The map is calculated.
- Double-click the map `Pit`. The Display Options dialog box is opened. Click OK.
- Check the result with the pixel information window.
- Close the pixel information window and the map window.

This results in a map with domain bool. If from all neighbors, the central pixel has the lowest value, that central pixel will be assigned True, or else False.

This calculation only allowed identification of pits in the Dem of only one pixel. Pits consisting of more pixels are not identified, nor are they corrected. In order to correct pits from the Dem, iterations should be used.

Iterations

Iterations are a special type of calculations. They are a successive repetition of a mathematical operation, using the result of one calculation as input for the next.

The calculation stops when the difference of the output compared to the input is negligible or if the number of steps is reached which was defined before. Iterations are always used in combination with neighborhood operations. Such an application might be for instance the selection of an item or area which fits a certain condition, starting from one pixel (for example the calculation of a flooded area, or the spreading of pollutant). The iteration functions available are listed in table 9.3.

Table 9.3: Iteration functions

MAPITER (startmap, iterexpr)	Performs iterations on the startmap according to the iteration expression until no pixel changes anymore.
MAPITER (startmap, iterexpr, times)	Performs a specified number of iterations on the startmap according to the iteration expression.
MAPITERPROP (startmap, iterexpr)	Performs iterations with propagation on a startmap; the newly calculated value for a pixel is used in calculating the next line instead of in the next iteration.
MAPITERPROP (startmap, iterexpr, times)	Performs a specified number of iterations with propagation on a startmap; the newly calculated value for a pixel is used in calculating the next line instead of in the next iteration.

After each iteration, ILWIS shows the number of changed pixels. This number is the total number of changes after performing one iteration in all directions (up, down, right and left).

Calculation of the flooded area, given dam site and dam altitude

In this example you are going to calculate the area that will be flooded after the construction of a dam in the mountains, North of Cochabamba.

The first step is determination of the dam site, dam altitude, freeboard and the designed water level *h*.

The planned dam is given in the raster map `Damsite`.

- Display raster map `Damsite`.
- Check the contents of the map.
- After that close the map window.

As you can see, the raster map `Damsite` contains the location of the dam, indicated with the word `Damsite`. The rest of the map has undefined values.

Now the `Damsite` map is combined with the raster map `Dem`, and a new map is created in which the `Damsite` is indicated with its height (3300 meters). The rest of the area shows the elevation of the terrain.

- Type the following formula on the command line of the main window:

  `Demnew:=IFF(isundef(Damsite),Dem,3300)` ↵
- Press OK in the Raster Map Definition dialog box.

  The map is calculated.

  Double-click the map `Demnew`. The Display Options dialog box is opened.
- Select the Representation: `pseudo`. Click OK.
- You will see that the `Damsite` now has the elevation 3300 meters. Check the result by clicking some pixels.

To determine the area to be flooded we need to create a map indicating one pixel in the future reservoir area. Using neighborhood operations this pixel acts as the

starting point for the calculation. You will use the pixel editor to create the new raster map.

☞

- From the File menu in the map window, select Create and Create Raster Map. The Create Raster Map dialog box is opened.
- Type for Raster Map Name: `Start`.
- Select the Georeference: `Cochabam`.
- Select domain: `value` with a Value Range between 0 and 1, and a Precision of `1`. Click OK.
- The Pixel Editor is opened and a message appears: Please zoom in to edit. Click OK. Zoom in on the `Damsite` until you see the individual pixels.
- Select a pixel just upstream of the Damsite, press the right mouse button and select Edit from the context-sensitive menu. Type the value `1`, and press Enter.
- Press the Exit Editor button. Now you have a raster map `Start` with 1 pixel that has the value 1. All other pixels have undefined values.
- Close the raster map `Demnew`.

Now you can perform the actual calculation of the flooded area. Iteration with propagation is used until there are no changes anymore in any of the pixel values. You can perform iterations by including the iteration statement in a Map Calculation formula. You can also use the dialog box of the Iteration operation.

☞

- Double-click the operation Iteration in the Operations-list. The Map Iteration dialog box is opened.
- Select the Start Map: `Start`.
- In the expression box type:
  `iff(Demnew>3280,Start,nbmax(Start#))`

This means: If the altitude in the new Digital Elevation Model is higher than 3280 m, then return the pixel values of raster map `Start` (which are undefined). Otherwise, assign the maximum value of the neighboring pixels found in raster map `Start` (which is a value of 1). In the first iteration there is only one starting pixel which has value 1. In every iteration, the neighboring pixels which satisfy the conditions (altitude < 3280 m) will get the same value as that starting pixel. This

will continue until next neighboring pixels have an altitude of more than 3280 m. (The upper side of the dam was at an altitude of 3300 m; the freeboard of the dam is 20 m; the actual water level is at 3280 m altitude).

☞

- Accept the default value for the Stop criterium: `Until No changes`. This means that the iteration continues until no changes occur anymore.
- Make sure that the option Propagation is selected, because then the new pixel value is immediately used in the calculation of the next line.
- Type for the Output Raster map: `Flooded`.
- Select the domain `value`, the Value Range `0` to `1` and the Precision of `1`.
- Select the check box Show.
- Click OK.

The calculation will take several minutes. The program will calculate in a downwards, upwards, left and right direction. In order to be able to do so, the map is rotated. When no changes occur after a full iteration, the final map is generated, and the Display Options dialog box is opened.

☞

- Accept the defaults and Click OK.
- Use the pixel information window and add the maps `Demnew`, `Start` and `Flooded`. Check the results.
- Close the pixel information window and the map window.

The output map `Flooded` can be used to calculate the volume of the water in the reservoir.

The concept of map iterations is a very powerful one. It allows you to create your own models for different types of applications. Consult the ILWIS Applications Guide, and the How to.. of the On-Line Help, for more examples of neighborhood operations.

Summary: Neighborhood operators

- Neighborhood operations are a special type of spatial analysis in ILWIS. They are calculations on pixels in which the outcome depends on the neighboring pixels.
- Neighborhood operations may be performed on user-selected pixels as well as on whole maps.
- Just as in filtering procedures, neighborhood operations make use of a filter. This imaginative window of 3 by 3 cells is moved over the raster map. Each cell of the output map is calculated according to the specified neighborhood expression.
- Iterations are a special type of calculation. They are a successive repetition of a mathematical operation, using the result of one calculation as input for the next.
- The calculation stops when the difference of the output compared to the input is negligible, or if the number of steps is reached which was defined before.
- Iterations are often used in combination with neighborhood operations. Such an application might be for instance the selection of an item or area which fits a certain condition, starting from one pixel (for example the calculation of a flooded area, or the spreading of pollutant).

9.3 Distance calculation

The Distance operation calculates the distance (in meters) from user specified source pixels to all other pixels in a raster map. The result of this operation is a raster map, where every pixel is assigned a value representing its distance to the nearest source pixel. The user-defined source pixels from which the distance is calculated may represent any kind of features, such as point features (for example rainfall stations), linear features (for example roads) or area features (for example a city block). The distance is calculated with a distance filter. The principle is shown in figure 9.3.

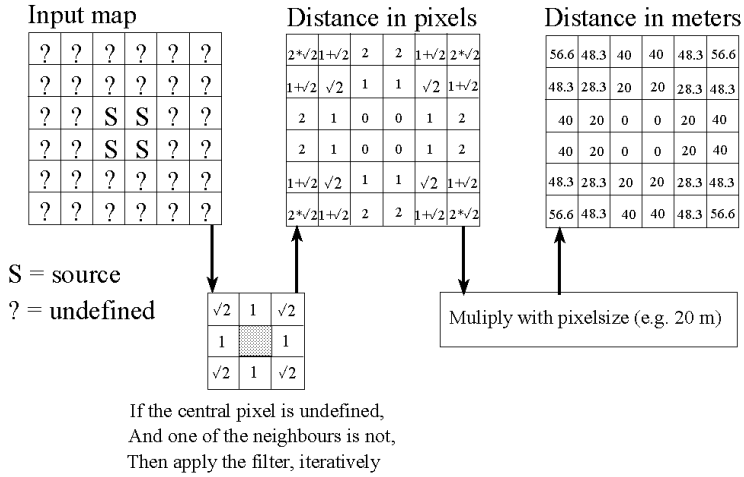


Figure 9.3: The principle of distance calculation.

A source map for distance calculation can be any raster map with domain type class, ID or value. This map should contain pixels with class names, IDs or values and pixels with undefined values. All pixels with a class name, IDs, or value in a source map, are considered as source pixels and distance values will be calculated for all pixels that are undefined. The distance filter will first calculate the distance in number of pixels. The distance from a source pixel to its horizontal vertical neighbors is 1, and the distance from a source pixel to its diagonal neighbors is the square root of 2. The filter will move first from the first pixel on the top line to the last pixel of the last line. Then, it will do the same operation in the reverse direction. Once the distance, in number of pixels, to the nearest source pixel is known, the final distance in meters is obtained by multiplying this with the pixel size. If not all parts in the source map are equally accessible, a so-called weight map can be used. A weight map contains pixels with values (weight factors) referring to the degree of accessibility and inaccessibility. In a weight map, accessible pixels are assigned positive values. A higher positive weight value for accessible pixels means that the area covered by these pixels (swamps, forest, bad

roads, etc.) can be crossed with more difficulty than other areas. Inaccessibility can be introduced in the weight map by negative weight factors. Pixels with a negative value (river, dens forest or impenetrable areas), represent those areas in the map which can not be passed. No distance value will be calculated for these pixels; the output value will be undefined.

By default, the output map (distance map) obtains the system Distance domain. The value range and step size of this domain can be adjusted each time you perform the Distance operation. You can also create or use a value domain of your own.

If line barriers, like a river, are used with a width of only one pixel (rasterized segments), these barriers should be broadened before performing the distance calculation, e.g. manually in the pixel editor or automatically with a Dilate filter or the Conn8to4 filter, which were used in exercise 9.1. This is because the distance program can pass diagonal barriers of one pixel wide, as it uses 8-connected distances (meaning each pixel has access to its eight neighbors: Horizontally, vertically, and diagonally).

In the following exercise three examples of distance calculations will be shown: One from line features (drainage), one from area features (city blocks, taking into account inaccessible areas) and one from point features (rainfall stations, using Thiessen polygons).

Simple distance calculation

The map `Drainage` with a class domain is used in order to calculate the distance from streams and lakes to any pixel in the map, under the assumption that all areas are equally accessible. The raster map `Drainage` contains pixels, representing the drainage (source pixels) and undefined pixel, representing the rest of the area in the map.

☞

- Double-click the operation Distance in the Operations - list.

The Distance Calculation dialog box is opened.

☞

- Select raster map `Drainage` in the list box Source Map.
- Type `Draindis` in the text box Output Map.
- Click the check box Show. Accept the other defaults by clicking the OK button.

The program will calculate first in a forward direction (starting from first pixel in the first line to the last pixel in the last line), after which it calculates in the backward direction. After a while the Display Options dialog box is displayed.

☞

- In the Display Options - Raster Map, select Representation `Clrstp12`.
- Change the Stretch range into `0` to `2000`.
- Click OK.

The distance map is displayed on the screen. This map is a value map, in which the pixel values refer to the distance of this pixel towards the nearest source pixels. You can also display the segment map `Drainage` in the same map window.

☞

- Drag the segment map `Drainage` to the map window.
- Click Ok in the Display Options dialog box.
- Enlarge the map and check the distance values by clicking on several pixels.
- Close the map window.

Calculating distance with weights: Time travel map

A time travel map is a distance map in which the calculated distance values in meters are converted to time units (seconds, minutes, or hours, etc.). For this process we need two raster maps as input: A source map and a weight map. The source map indicates from where the distance should be calculated, and the weight map indicates which areas are more difficult to cross.

The following examples show how to convert distance values to time units, assuming an average speed of 20 km/h:

– When the output distance values are divided by 20000 (m/h), the time in hours to reach the nearest source pixel is obtained.

– When the output distance values are multiplied by 60 and divided by 20000, the time in minutes to reach the nearest source pixel is obtained.

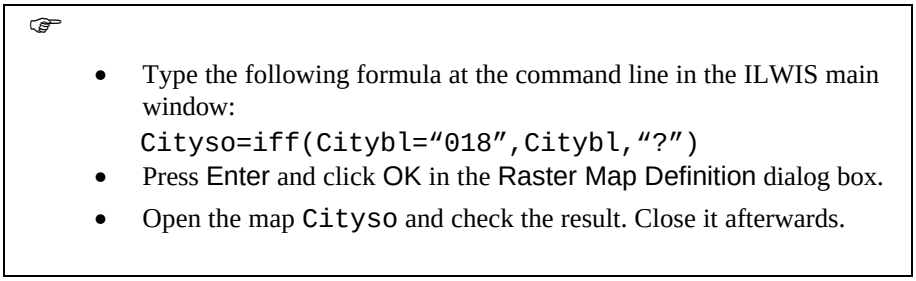
– When the output distance values are multiplied by 60*60 and divided by 20000, the time in seconds to reach the nearest source pixel is obtained.

In this exercise, the city block map of Cochabamba (`Citybl`) is used to calculate how much time it takes from any part in the city to reach the central plaza ("Plaza 14 de Septiembre"). For reasons of simplicity we assume that you can travel equally

fast in all the streets of the city. The city blocks themselves cannot be crossed, with the exception of those with a recreational use (parks, etc.), which can be crossed with an average velocity of 5 km/hour.

First step: Creation of the source map

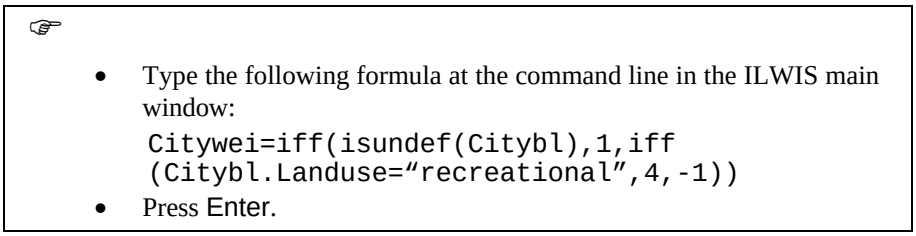
One of the input maps required for this operation, is a source map in which only the central plaza has a value and the rest of the map has undefined values. The central plaza has the ID "018" (you can check that if you like).



- Type the following formula at the command line in the ILWIS main window:
  `Cityso=iff(Citybl="018",Citybl,"?")`
- Press Enter and click OK in the Raster Map Definition dialog box.
- Open the map `Cityso` and check the result. Close it afterwards.

Second step: Creation of the weight map

The second input map required for this operation, is a weight map where all the pixels, except the source pixels, are assigned weight factors to simulate the difficulty of crossing pixels, which form barriers such as building blocks, the river, the lake, mountain ridges, etc. In this exercise the roads are assigned a value 1, the recreational areas a value 4 and the rest of the city blocks will receive the value -1 (negative weights make the pixels inaccessible). Note that the weight factors are inversely proportional to the speed. In this exercise the average speed (20 Km/h) is set to weight factor 1 (along the streets) and to obtain the minimum speed of 5 km/h in the recreational areas we use a weight factor 4.



- Type the following formula at the command line in the ILWIS main window:
  `Citywei=iff(isundef(Citybl),1,iff (Citybl.Landuse="recreational",4,-1))`
- Press Enter.

In this formula first the roads (which have undefined values) are assigned a value 1. Then, if the land use type (in the column `Landuse` of the table `Citybl`) is "`recreational`", the resulting value is 4. All other pixels (the other city blocks) will get a value -1.

Third step: Calculating distances

☞

- Double-click the Distance operation in the Operations - list.

The Distance Calculation dialog box is opened.

☞

- Select raster map `Cityso` in the list box Source Map.
- Click the check box Weight Map.
- Select raster map `Citywei` in the list box Weight Map.
- Type `Citydist` in the text box Output Map.
- Click the check box Show.
- Accept the defaults by clicking the OK button.

The distance calculation takes quite some time. The program calculates forward (from left to right and from the first line to the last line) and backward (the reverse). The tranquilizer shows the number of changes that are made, as well as the line which is being calculated. The percentage that is shown is the percentage of the map. It does not indicate how much percentage of the total calculation is finished. The calculation is done iteratively until there are no more changes. When the calculation is finished, after approximately 10 minutes, the Display Options - Raster Map dialog box is displayed.

☞

- Select the `Pseudo` representation and click OK. The weighted distance map is displayed on the screen.
- Find out the distance from the airport to the city center.

Fourth step: Converting distances to travel time

Now that the weighted distances from each part in the map towards the central plaza are known, we only need to convert the distance values (in meters) to time units (minutes). The average speed in the city is assumed to be 20 km/h in the streets and 5 Km/h in the parks. To create the travel time map:

☞

- Type the following formula on the command line:
  `Citytt=(Citydist*60/20000)`↵
- Display the map `Citytt` with the `pseudo` representation.
- Check how much time it takes to travel from the airport to the center.
- Close the map window when you are finished.

Calculating distances: Thiessen map

In the third and last example of distance calculation, you will have a look at a Thiessen map. In a Thiessen map each pixel is assigned the class name, identifier, or value of the nearest point. For example, schools, hospitals, water wells, etc. can be represented by points. The output of a nearest point operation on such a point map, gives the 'service area' of the schools, hospitals or water wells, based on the shortest distance (as the crow flies) of a point and pixels.

A Thiessen map will be calculated from the point map `Rainfall`, containing -fictitious- rainfall stations in the Cochabamba area. Before we can calculate it we first need to rasterize the point map (`Rainfall`).

☞

- Click with the right mouse button on the point map `Rainfall` and select Rasterize and the command Points to Raster from the context-sensitive menu.

  The Rasterize Point Map dialog box is opened.
- Select the georeference `Cochabam`. Select Show. Click OK.
- Click OK in the Display Options dialog box.

  The map is now displayed. Since each point is represented by one single pixel the map appears to be empty. Only if you zoom in a lot, you can see the rasterized points.
- Close the map.
- Double click the operation Distance in the Operation - list.

The Distance Calculation dialog box is opened.

☞

- Select raster map `Rainfall` in the list box Source Map.
- Type `Raindist` in the text box Output Map.

- Click the check box Show.
- Select the check box Thiessen Map and enter the name: `Rainarea`.
- Click OK.

  The distance calculation takes a few minutes. After that the Display Options dialog box is opened.
- Click OK. The distance map `Raindist` is displayed.
- Open the Thiessen map `Rainarea`.
- Click the various units to find out their meaning.

The Thiessen map shows the areas which are nearest to each one of the rainfall stations. We can also display the attribute data (rainfall values) instead of the rainfall stations. We will first calculate the total rainfall per year for each of the stations.

☞

- Open the table `Rainfall`.
- Type the following formula on the command line of the table window:

  ```
  Raintot = January + February + March + April + May + June +
  July + August + September + October + November + December
  ```
- Press Enter. The Column properties dialog box is opened.
- Click OK.
- Close the table window and activate the map `Rainarea`.
- Click with the right mouse button inside the map and select the last line (`1 map Rainarea`) from the context-sensitive menu.
- Select the Check box Attribute and select the column `Raintot`.
- Select the representation `Pseudo` and click OK.

  Now the total rainfall per year in shown for each of the Thiessen areas.
- Close the maps `Raindis` and `Rainarea`.

The Thiessen map could be used to calculate the total rainfall per year for each catchment of the map `Catchmen`.

☞

- Calculate the annual rainfall per catchment. First use the Attribute map operation to generate a map `Raintot` from the Thiessen map

Rainarea, and use column Raintot from the attribute table Rainfall.

- Use the Cross operation to overlay the maps Catchmen and Raintot.
- Calculate the total rainfall per catchment from the cross table with the aggregation function Sum in the table window.
- When you have finished, close all the maps and table windows that are still open.

A Thiessen map can also be created, without the need for calculation a distance map, using the Nearest Point interpolation method (see chapter 11).

Summary: Distance calculation

- The Distance operation calculates the distance (in meters) from user specified source pixels to all other pixels in a raster map.
- All pixels with a class name, IDs, or value in a *source map*, are considered as source pixels and distance values will be calculated for all pixels that are undefined.
- A weight map contains pixels with values referring to the degree of accessibility. Negative weights indicate inaccessibility, and positive values, higher than 1, indicate areas that can be crossed with more difficulty.
- The units in a distance map are meters. They can be converted with Map Calculation formulas into time.
- A by-product of distance calculation is a Thiessen map, in which the 'service area' of source pixels, based on the shortest distance is indicated. Thiessen maps can also be made from point maps using the Nearest point interpolation.

9.4 Areanumbering

The objective of area numbering, is to assign *unique identifiers* to those areas in a raster map that are having the same value, and that are connected. The operation is based on a moving 3 by 3 window which assigns a value to the center pixel. The connected areas recognized by the window are either individual pixels, or a set of pixels with the same class/ID or value, which may be either 4-connected (only those connections that are in horizontal or vertical direction count) or 8-connected (also diagonal connections are taken into account). Figure 9.4 shows a simple example of the area numbering operation applied to a bit map.

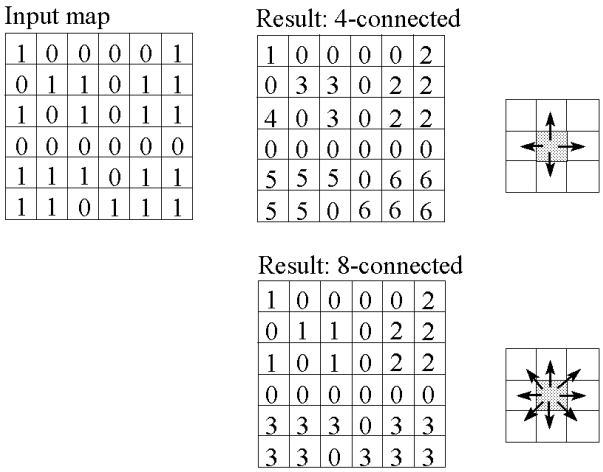


Figure 9.4: Simple example of Area numbering with a bit map as input. The pixels which are connected are assigned the same code. Different results are obtained when only the horizontal and vertical neighbors are considered (4-connected) or whether all neighbors are considered (8-connected)

Area numbering may be a useful tool in the following situations:

- If you want to perform a Cross operation with another map and you want to distinguish each connected area separately, instead of using mapping units.
- If you want to add special attribute data to each connected area, instead of the mapping units.

Each connected area (areas consisting of pixels with the same value, class name, or ID) is assigned a unique pixel value such as Area 1, Area 2, etc. Furthermore, an attribute table is created, which contains the new Area IDs and the original class names, IDs or values.

In this exercise, the Area Numbering operation is performed to the map `Geom` , which has a class domain.

☞

- Double-click the Area Numb operation in the Operation - list. The Area Numbering dialog box is opened.
- Select raster map Geom in the list box Raster Map.
- Type Geomid in the text box Output Raster Map.
- Click the check box Show.
- Accept the defaults and click the OK button. The Display Options - Raster Map dialog box is opened.
- Click the option button 31 colors.
- Click OK. The map is displayed. Check the meaning of the mapping units.
- Double-click a unit. The Edit Attribute dialog box is opened, in which you can also read the geomorphologic class name.
- Close the map window when you are finished the exercise.

Summary: Area numbering

- The objective of area numbering, is to assign *unique identifiers* to those areas in a raster map that are having the same value, and that are connected.
- Area numbering may be a useful tool in the following situations:

  If you want to perform a Cross operation with another map and you want to distinguish each connected area separately, instead of using mapping units.

  If you want to add special attribute data to each connected area, instead of the mapping units.
- Area numbering is fairly similar to the UniqueID operation that can be used for point, segment and polygon maps.
- The input map for area numbering should not contain values that are different for every pixel. Before performing AreaNumbering on value maps with many different values, first classify or smooth the input map. To prevent that inexperienced users will try to do AreaNumbering on a value map with many different values, you cannot select value maps as input in the AreaNumbering dialog box. However, experienced users can use value maps as input by using an expression on the command line (see chapter 12).

9.5 Connectivity operations

Connectivity operations look at spatial units that are connected (using a set of pre-defined rules). These spatial units may either occur in raster or vector maps. This group of operations can be subdivided into:

- *Contiguity functions*: Connected areas that share (a) common characteristic(s).
- *Proximity functions*: Connected areas having the same distance (in time, distance, costs, etc.) to a given point, line or area.
- *Network functions*: Areas (usually lines) that form a set of interconnected features through which resources from one location to another are moved.
- *Spread functions*: Connected areas that result for spreading, dilution or accumulation of phenomena from a given point, line or area, and;
- *Seek functions*: Connected areas (usually lines) that form an optimum pathway according to (a) specified decision rule(s).

Contiguity operations

Contiguity operations examine the connected areas that share (a) common characteristic(s). In order to know which areas are connected, topological information is required. In general, these operations are applied to identify (retrieve) areas with a specific size and with specific characteristics.

In this exercise we will show you how you can use the information of neighboring polygons to solve the following problem:

Find out a forested area which is not bordering agricultural land or the city, and which has a closed area of at least 200 hectares.

In order to solve this problem we need to know the following:

- Which one of the forested areas has no borders with agricultural or urban areas?
- Which one of the forested areas is at least 200 hectares large?

In order to solve these two sub problems we need to work with the polygon map Landuse. The Landuse map has a class domain. This means all the polygons with the same land use have the same class. However, we cannot work on mapping units. We have to know the information for each individual polygon. In order to do so we apply the operation Unique ID. This operation will transform a class map into a unique identifier map, in which each polygon receives a single code.

- Double-click the Unique ID operation from the Operations - list.

The Unique Id dialog box is opened.

☞

- Select the polygon map `Landuse` as the Input Map.
- Create an identifier domain `Landid`.
- Type `LandID` as the Output Map. Click Show.
- Type the following description: `Unique Land use polygons.`
- Click OK. The map is being calculated.
- Click OK in the Display Options - Polygon Map dialog box. The map is now displayed.
- Find out the codes of the map by clicking on the units.
- Double-click on a unit. Now the land use type for each polygon is displayed in the Edit Attributes window.

Apart from the map `LandID`, a table with the same name is also generated in which the land use type for each polygon is stored. The next thing we need to know is which are the neighboring polygons of each polygon. For this you can use the operation NeighbPol. This operation will calculate for each polygon in the map, what are its neighbors, and what is the length of their boundary lines.

☞

- Close the map `LandID.`
- Double-click the NeighbPol operation from the Operation-list.

The Neighboring Polygons dialog box is opened.

☞

- Select the polygon map `LandID` as the Input Map.
- Type `LandIDnb` as the Output Table.
- Click Show.
- Type the following description: `Neighbors of the unique land use polygons.`
- Click OK. The table is opened after it has been calculated.

The result of this operation is a table (`LandIDnb`) containing three columns:

– `PolName1`: The polygon for which the neighbors are calculated.

– PolName2: The polygon neighboring the polygon shown in PolName1.

– Length: The length of the boundary line between the two polygons.

The combination between the polygons in PolName1 and those in PolName2, represent the polygons located at the margins of the map. The undefined values (?) refer to the boundary of a polygon located at the margin of the study area.

We are interested in the polygons with the land use type Forest in the column PolName1. As you can see, we do not have information yet on the land use types (only on the polygon ID's). This information exists, however, in the table LandID, in which for each ID the land use is given. We will have to join the two tables in order to read the information of the land use in table LandIDnb.

Joining of tables was already discussed extensively in chapter 5.

- Open the Columns menu and select the Join command.

The Join dialog box is opened. Since the column PolName1 has the same domain as the table LandID, the joining is fairly simple.

- Select the Table : LandID.
- Select the Column: Landuse.
- Select the Key-column: PolName1.
- Type the Output Column name: Landuse1.
- Click OK in the Join dialog box.
- Click OK in the Column Properties dialog box.

Now we know for each of the polygons ID's in column PolName1 what the land use is. We also need to know the land use of their neighboring polygons, listed in column PolName2.

- Do the same Join operation, but now select the key-column PolName2 and type the Output column: Landuse2.

Now that we know the land use type of each polygon and its neighbors, we can start to evaluate which one of the forest polygons has no borders with agricultural or urban polygons. We can do this with a calculation formula. There are four types of land use which we want to exclude: `Agriculture`, `Agriculture (irrigated)`, `Urban center` and `Urban periphery`. To simplify the formula, we already know that none of the forested area is bordering the urban area. The formula would become very large if we would have to type the full names. Fortunately we can use a shortcut. If we use the function Left(column,length) in the following way:

– `Left(Landuse2,2)="ag"`, this will incorporate both the land use types `Agriculture` and `Agriculture (irrigated).`

☞

- Locate the mouse pointer on the command line of the table window and type the following formula:
  `Good=(Landuse1="forest") and (left(Landuse2,2)<>"ag")`
- Press Enter.
- Click OK in the Column Properties dialog box.

The column good has a bool domain. You know now all the polygons with the land use type "`Forest`", that are not bordered by agricultural or urban polygons.

☞

- Open the Columns menu.
- Select Sort.
- Select the column: `Landuse1`.
- Scroll down the table until you see the records with the land use type "`Forest`" in the column `Landuse1`.
- Check the result of the formula.

As you can see there are 3 forest polygons which are bordered by agriculture. Another 13 are bordered by either grassland or shrubs, and one forest polygon is located at the margin of the area. However, these records may refer to the same forest polygons, which may be bordered by agriculture on one side and grassland on the other. Therefore, we should take the minimum value of the column `Good` in the next step.

Now we know which of the forest polygons is not bordered by agricultural or urban land.

☞

- Close the table `LandIDnb` and open the table `LandID.`
- Open the Columns menu and select the Join command. The Join Column dialog box is opened.
- Select the Table: `LandIDnb` and select the Column: `Good`.
- Deselect the check box Key-column. Select the check box Aggregate.
- Select the Function: `Minimum`. Select Group by: `PolName1`.
- Type the name of the output column : `Good`. Click OK.

Now the table `LandID` contains the column `Good`, that indicates with a value 1 those polygons that have forest, and have a neighbor polygon which is not agriculture nor urban.

The calculation for the first requirement is finished. Now we still need to know which of the forest polygons has an area larger than 200 hectare. We can obtain this information from the polygon histogram.

☞

- Click with the right mouse button on the polygon map `Landid` and select Statistics, and Histogram, from the context-sensitive menu. The Calculate Histogram dialog box is opened.
- Click OK. The Polygon Histogram is shown as a table.

Now that the area of the polygons is known, we can now make the final evaluation.

☞

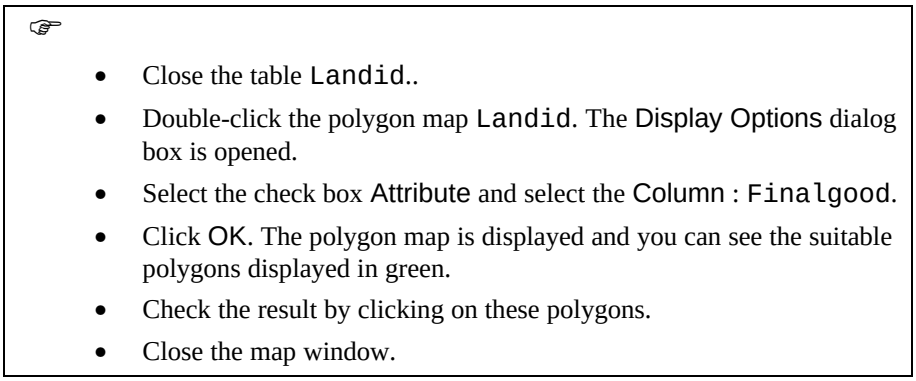
- Close the Polygon Histogram `Landid`.
- Activate the table `Landid`.
- Type the following formula on the command line of the table window:

  ```
  Finalgood:=(good=1) and
  (Landid.hsa.area>2000000)
  ```

  Press Enter. Click OK in the Column Properties dialog box.

The calculation uses the column `Area` from the Polygon histogram table (extension .hsa). Those polygons which contain the word "`True`" in the column `Finalgood`, fulfill the requirements stated in the beginning. They are forest polygons, with an area of at least 200 hectares, and not bordered by agriculture or city polygons.

The final task is to display these polygons.



- Close the table `Landid`..
- Double-click the polygon map `Landid`. The Display Options dialog box is opened.
- Select the check box Attribute and select the Column : `Finalgood`.
- Click OK. The polygon map is displayed and you can see the suitable polygons displayed in green.
- Check the result by clicking on these polygons.
- Close the map window.

Summary: Connectivity operations

- Connectivity operations look at spatial units that are connected (using a set of pre-defined rules).
- The operation Unique ID is used to transform a class map into a unique identifier map, in which each polygon receives a single code.
- The operation NeighbPol calculates for each polygon in a map, what are its neighbors, and what is the length of their boundary lines.
- The resulting table can be used for connectivity operations.